

this enables users to perform operations and use applications such as calendars, address books, chat rooms, data processors, etc. as part of a distributed system.

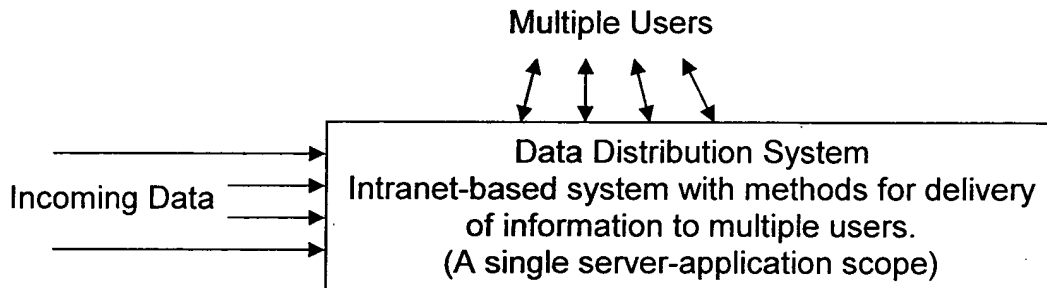
In addition, claim 1 requires that the DOTS include system methods that enable “addition of new system elements and modification of the functionality and content of existing system elements” (claim 1, lines 10 - 13). In other words, the DOTS include system methods that enable processes and services (as well as data) to be added and modified across the entire network to modify the way the system itself functions or to provide new functionality in a manner that is transparent to the user. Here again, Grasso et al provide for distribution of new and modified documents, but not new and modified process and services. Similar limitations are present in both of the remaining independent claims (i.e., claim 17, lines 8 - 9; and claim 21, lines 8 - 9 and 11 - 13).

The system disclosed by Grasso et al. differs in a fundamental way from the present invention. Grasso et al. teach a system with a single centralized server to which multiple clients connect. The present system, on the other hand, teaches a distributed communications model between a plurality of DOTS communicating over a network. This enables different systems to share data with each other by using the same methods for different data types and applications, and to search other systems.

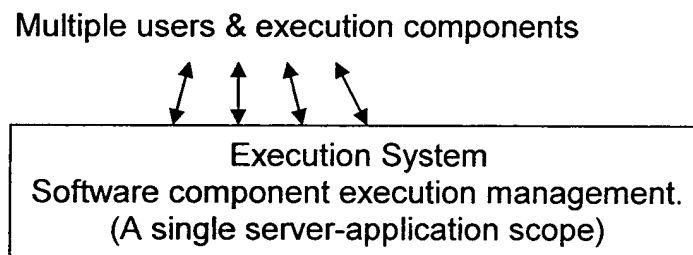
Helland et al. Helland et al. have been cited as disclosing a plurality of DOTS having a plurality of system elements and system methods. In response, Applicant notes that Helland et al. describe a “software component execution management using context objects for tracking externally-defined intrinsic properties of executing software components within an execution environment.” The system describes how software components are executed within a *single* server, and in particular the described methods are implemented in Microsoft Windows Server software. In contrast, the present application discloses multiple systems (DOTS) that distribute data and provide services (executions), but the present invention focuses *on the interaction between multiple DOTS communicating over a network*. This limitation is set forth in each of the independent claims (i.e., claim 1, lines 4 - 5; claim 17, lines 4 - 5; and claim 21, lines 4 - 5).

The following diagrams highlight the differences between Grasso et al., Helland et al. and the present invention:

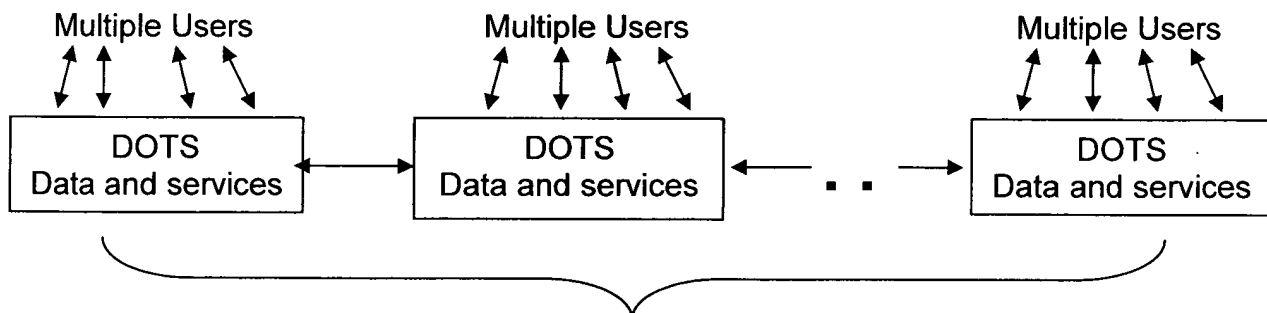
Grasso et al. -



Helland et al. -



Present Invention -



Distributed set of applications and systems (DOTS) in which multiple systems interact, adjust each others' behavior, and share data and services across multiple applications.

To summarize, Grasso et al. and Helland et al. disclose applications that use a single server to serve multiple users. The present application describes a distributed set of applications and systems (DOTS) communicating over a network that allow multiple systems to interact, adjust each others' behavior, and share data and services across multiple applications.

Dependent Claims. With regard to dependent claims 2, 3 and 6, Applicant notes that these claims require user evaluations or usage value summaries associated with system elements. Dynamically ascribing values to system elements is neither taught nor suggested by Grasso et al. or Helland et al. For example, such values could be used as a type of virtual currency to enable users to purchase elements of given values with an equal amount of currency. A user who contributes an element that subsequently becomes highly valued would receive that amount of currency, and be able to use it to purchase other system elements.

As noted on page 2 of the Office Action, Grasso et al. suggest the idea of user evaluation of data distributions. However, this is only in the context of a single-server application. As previously discussed, nothing in Grasso et al. or Helland et al. teaches or suggests a plurality of DOTS communicating over a network.

Regarding claims 2-5, the distribution policy described by Grasso et al. specifies delivery of information to a user similar to an email distribution system that provides priority and security markers that become a user's properties. In contrast, the present application teaches a *distributed multi-server system* providing "access to system elements with the DOTS, and generating access negotiation requests to other DOTS based on user and group privileges and system element properties." This is substantially different from Grasso et al and Helland et al., as Grasso et al. teach a distribution system with methods that distribute data to multiple users based on privilege properties, and Helland et al. teach execution methods that take into account the user's privileges. The present application, on the other hand, teaches a community of interacting DOT systems that permits different DOTS to negotiate

system element trade based on user, group, or system access privileges and system element access type and usage value.

Helland et al. have been cited as teaching access control of component-based server applications using roles. Helland et al. move role-based access control from the database where it is usually presented to the application level. Each role is directly mapped to a set of privileged data access functions or services provided by a specific server application. In contrast, the present application teaches a distributed multi-server system where every data object as well as every service has its own security type as its property. The security type life cycle is not limited by a single application. Security types represent another dimension of mapping privileges. This allows more flexible 3-dimensional mapping across roles and security types where each role can represent a collection of multiple users based on the users' properties and each security type can represent a collection of system elements (data or services) based on the system element properties. System elements and users can belong to different systems (DOTS) and have different life times. Privileges are located on cross-sections of this 3-dimensional matrix (see Fig. 8). Here, multiple applications use security types as another dimension while mapping application specific roles to data and services that can be shared among the distributed applications. The access map is a 3-dimensional map (see Fig. 8) versus the usual 2-dimensional map by Helland et al. that only makes sense within a single application.

With regard to claims 9 - 10 and 17 - 23, nothing in Grasso et al. or Helland et al. teaches or suggests a thematic search controller to search for system elements with selected parameters, or to search system elements across a plurality of DOTS. Both of these references involve only single server-based applications, as previously discussed.

Dependent claim 13 requires a "repeated actions scheduler enabling users to schedule periodic system operations." Grasso et al. disclose scheduled data distributions. However, nothing in Grasso et al. teaches or suggests scheduling the other types of the system operations that DOTS are capable of providing, including services across multiple servers.

Dependent claim 14 requires an "action object defined by an executing environment descriptor and an action statement executable in this environment." Here again, nothing in Grasso et al. or Helland et al. teaches or suggests this element, particularly in the context of services shared across multiple servers.

Dependent claim 15 requires "a remote control scenario defined by a set of remote action objects", which is necessarily executed across multiple systems. Nothing in Grasso et al. or Helland et al. teaches or suggests a remote control feature. This feature enables a client or DOTS to control and launch processes in another DOTS. Thus, users can control documents and applications on other systems, enabling new aspects of telecommuting. For example, the system might be used to install an application that would set an alarm on a remote machine, scheduled to go off when a critical state is reached. This application could then use the remote-control feature to send a message or run an application on a given user's system in order to notify the user.

Helland et al. teach "properties needed to run the component in the execution environment" and related privileges interactively assigned by a user to define who can execute this component. However, there is a fundamental difference between Helland et al. and the present invention. Helland et al. describes a process of building a package of components. This package of components is a pre-built application library ready to run. This is the process of building software, creating an application engine. In contrast, the present application discloses a remote scenario that uses such libraries to create different applications on-the-fly. The scenario uses the application engines of multiple systems as instruments in the concert to play new music each time. Each remote scenario element is a note in the music that points to one of the application components in one of the systems. Users can use existing instruments to create multiple performances without building new software, just by naming the proper elements and providing specific conditions. No compilation or assembly work is needed. DOTS are ready to play scenarios; they can read the sheet music.

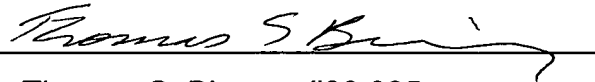
Favorable reconsideration is respectfully requested.

Respectfully submitted,

DORR, CARSON, SLOAN, BIRNEY & KRAMER, P.C.

Date: July 25, 2005

By: _____



Thomas S. Birney #30,025
3010 East 6th Avenue
Denver, Colorado 80206
(303) 333-3010

Attorneys for Applicant